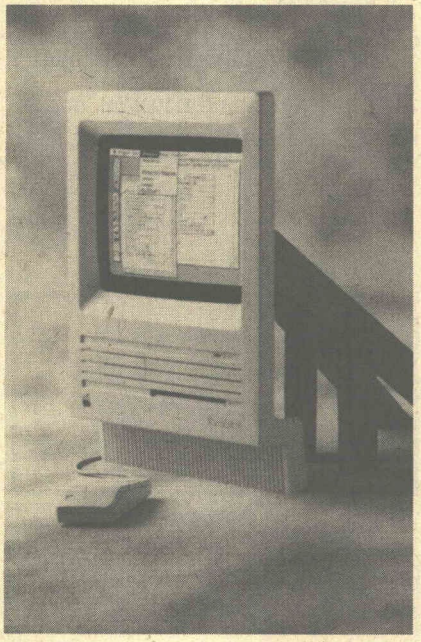
ศรันย์ ไมตรีเวช

ออปเจ็กต์โมเดลของ เอ็กเซล

เสนอออปเจ็กต์โมเดลที่มีความสามารถที่สุดในโลกแห่งการพัฒนาแอพพลิเคชันบนวินโดวส์และแมคอินทอช



เอ็กเซล 5.0 มีออปเจ็กต์ที่สามารถโปรแกรมได้ทั้งหมด 128 ตัวออปเจ็กต์เหล่านี้จะถูกจัดเรียงแบบลำดับชั้น (Hierarchical Order) ออปเจ็กต์แต่ละตัวจะมีทั้งพรอพเพอร์ตี้ส์และเมธอดมากมาย ซึ่งมากเท่าไหร่ก็หมายความว่าคุณยิ่งใช้งานมันได้หลากหลายขึ้นเท่านั้น

เรานำออปเจ็กต์เหล่านี้ไปรวมไว้ในแอพพลิเคชันที่ทำขึ้น ด้วยการใช้วิชวลเบสิกสำหรับแอพพลิเคชันในการป้อนค่าหรือตัวกำหนด (settings) ให้กับพรอพเพอร์ตี้หรือไม่ก็เรียกใช้เมธอด คงพอนึกภาพการเขียนโค้ดออกนะครับสำหรับผู้ไม่เคยสัมผัสวิชวลเบสิกมาก่อน เรื่องของเรื่องก็คือคุณจะใช้ออปเจ็กต์สักตัว คุณต้องเข้าไปเกี่ยวข้องกับพรอพเพอร์ตี้ส์หรือเมธอดของมันไม่ทางใดก็ทางหนึ่ง อยู่ ๆ อยากเรียกใช้มันแล้วอ้างชื่อมันลอย ๆ อย่างเดียวมันก็ทำอะไรให้ไม่ถูก เหมือนคนงานที่เป็นแต่รับคำสั่งจากหัวหน้าให้ทำโน่นทำนี่ ต้องระบุอย่างชัดเจนว่า “นาย ก. วิ่ง” หรือ “ นาย ข. เทน้ำ” อย่างนี้นาย ก. หรือนาย ข. ถึงจะรู้หน้าที่ว่าเขาสั่งให้ตนทำอะไร ต้องเข้าไปเกี่ยวข้องกับวัตถุหรือบุคคลอื่นใดหรือไม่ แต่ถ้าจู่ ๆ เรียกกันลอย ๆ ว่า “นาย ค.” อย่างนี้นาย ค. ก็ได้แต่รับรู้ว่ามีการเรียกชื่อตนแล้วก็จะต้องหันมาถามกันให้ยาวยืดต่อไปอีกว่า “เรียกผมทำไม?”

เอ็กเซลได้เสนอออปเจ็กต์โมเดลอันทรงความสามารถที่สุดในโลกแห่งการพัฒนาแอพพลิเคชันทั้งบนวินโดวส์และแมคอินทอช มาดูรายละเอียดในฉบับนี้กันเลยครับ

1. วิชวลเบสิกสำหรับแอพพลิเคชัน

ผมรู้ครับว่าหลายคนกำลังอยากรู้เรื่องเกี่ยวกับออปเจ็กต์ของเอ็กเซล ก่อนที่เราจะเข้าไปดูกันจริง ๆ ขอให้มาทำความเข้าใจเกี่ยวกับวิชวลเบสิกสำหรับแอพพลิเคชันกันหน่อย

อย่างที่ได้กล่าวไปแล้วเมื่อฉบับก่อนว่า แมคโครของเอ็กเซล 5.0 นี่เป็นโมดูลต่างหากจากตัวสเปรดชีต โดยที่วิชวลเบสิกจะถูกแทรกเข้าไปในเวอร์กบุ๊ค (workbook) ตัวหนึ่ง ด้วยวิธีเลือก

คำสั่งแมคโครจากเมนู Insert แล้วเลือกโมดูลที่ต้องการ

แมคโครชนิดใหม่นี้มีอยู่สองชนิด คือ โพรซีเดอร์กับฟังก์ชัน ความแตกต่างระหว่างสองอย่างนี้ก็เป็นเรื่องของการคืนค่านั่นแหละครับ หากเป็นฟังก์ชันจะคืนค่า แต่ถ้าเป็นโพรซีเดอร์ก็จะไม่คืน

จุดใหญ่ที่เราจะโฟกัสกันคือ เรื่องแมคโครที่ไม่คืนค่า ซึ่งก็ได้แก่ แมคโครที่เป็นโพรซีเดอร์นั่นเอง ทุกโพรซีเดอร์จะขึ้นต้นด้วยคีย์เวิร์ด Sub และปิดท้ายด้วยคีย์เวิร์ด End Sub ตัวอย่างข้างล่างนี้เป็นคำสั่งให้เกิดเมสเสจบ็อกซ์ขึ้นมา

```
Sub Say_Hello ()
        MsgBox "สวัสดีจ้า!"
End Sub
```

สังเกตนะครับว่าเราขึ้นต้นกันด้วยคำว่า Sub และต่อด้วยชื่อโพรซีเดอร์ของแมคโคร ซึ่งข้างบนนี้ตั้งไว้ว่า "Say_Hello" การรันวิชวลเบสิกสำหรับแอพพลิเคชันทำได้โดยการเลือกคำสั่งแมคโครจากเมนู Tools แล้วเลือกชื่อแมคโครที่คุณต้องการ และต้องบอกเสียหน่อยว่าจริง ๆ แล้ววิธีรันอื่นยังมีอยู่คือ สิ่งที่เราจะเรียกมันว่า "เหตุการณ์" (Events)

ตัวอย่างสามัญที่สุดน่าจะได้แก่ เหตุการณ์ "คลิ๊ก" ซึ่งเกิดการกระทำขึ้นได้โดยเอาโค้ดเข้าไปใส่ไว้ในส่วนโค้ดที่เกี่ยวข้องกับการ "คลิ๊ก" ของปุ่มบนเวอร์กชีตซึ่งผู้ใช้จะทำการติดต่อด้วย

ก่อนอื่นคุณต้องสร้างปุ่มขึ้นมาด้วยการเลือก Create Button บนทูลบาร์ จากนั้นก็คลิ๊กปุ่มเม้าส์ขวาบนรูปปุ่มที่สร้างขึ้น แล้วเลือกคำสั่ง Assign Macro to Object จากเมนูลัด (shortcut menu)

ตัวแปร

แง่หนึ่งที่สำคัญของวิชวลเบสิกสำหรับแอพพลิเคชันคือ ตัวแปรและการดีแคลร์ตัวแปร วิชวลเบสิกสำหรับแอพพลิเคชันเสนอฟีเจอร์ใหม่ซึ่งจะไม่บีบบังคับให้คุณต้องดีแคลร์ตัวแปรเสียก่อนแล้วถึงจะใช้ได้ อย่างเช่น ตัวอย่างข้างล่างนี้เป็นการป้อนค่า 1 เข้าไปไว้ในตัวแปรอินทีเจอร์ชื่อ int_one ถึงแม้ int_one จะไม่เคยถูกดีแคลร์ไว้เลยในส่วนใด ๆ ของโปรแกรม

```
Sub Assign_Value ()
        Int_one = 1
        Range("A1").Value = Int_one
End Sub
```

ถึงจะมีฟีเจอร์นี้ (ซึ่งอันที่จริงก็ไม่ได้ใหม่เอี่ยมอ่องเท่าไหร่ เราเคยเห็นมาก่อนแล้วในเบสิกโบราณ) เราท่านก็คงไม่เอาไปใช้กันเท่าไหร่หรอกใช่ไหมครับ โดยเฉพาะถ้าต้องเขียนโค้ดยาว ๆ ใช้ตัวแปรเยอะ ๆ ขืนใช้เรื่อยเปื่อยโดยไม่มีการดีแคลร์ไว้ก่อนจะปนกัน ไม่ใช่เรื่องสนุกนักถ้าเราต้องมานั่งแก้บักบ่อย ๆ เพราะพิมพ์ชื่อตัวแปรผิดหรือใช้ในที่ที่ไม่ควรใช้ ฟีเจอร์นี้เหมาะกับโปรแกรมสั้น ๆ ซึ่งเผอิญเป็นไปได้มากกับการเขียนแมคโครให้เอ็กเซลนี้ ถ้าแอพพลิเคชันของคุณไม่มีอะไรซับซ้อนยุ่งยาก การไม่ต้องดีแคลร์ตัวแปรเป็นหนทางหนึ่งของความสะดวกสบายในการเขียนโค้ด นึกอยากใช้ตัวแปรเมื่อไหร่ก็ตั้งชื่อกันสด ๆ เดี๋ยวนั้นเลย

วิธีบังคับให้โค้ดต้องมีการดีแคลร์ตัวแปรทำได้สองวิธี คือ

1. เลือกคำสั่ง Options จากเมนู Tools ในโมดูลเขียนโค้ด แล้วเลือก Require Variable Declaration จาก Moule General Tab

2. พิมพ์คำว่า Option Explicit ที่ส่วนบนสุดของโมดูลเขียนโค้ด

การดีแคลร์ตัวแปรทำได้โดยใช้คีย์เวิร์ด Dim (ซึ่งย่อมาจาก Dimension) ตามด้วยชื่อตัวแปรที่คุณต้องการ คุณสามารถระบุชนิดตัวแปรได้โดยใช้คีย์เวิร์ด As หลังชื่อของมัน แล้วตามด้วยชนิดที่ต้องการ

จากตัวอย่างโค้ดที่ผ่านมา ถ้าหากคุณต้องการให้โค้ดบังคับการดีแคลร์ตัวแปร ก็จะเขียนออกมาได้ว่า

```
Sub Assign_Value ()
        Dim Int_one as Integer
        Int_one = 1
        Range("A1").Value = Int_one
End Sub
```

ตัวแปรที่จะใช้ในแมคโครมีอยู่หลายชนิดด้วยกันดังต่อไปนี้

- Boolean (2 ไบต์)
- Integer (2 ไบต์)
- Long (4 ไบต์)
- Single (4 ไบต์)
- Double (8 ไบต์)
- Currency (8 ไบต์)
- Date (8 ไบต์)
- Object (4 ไบต์)
- String (1 ไบต์ต่อหนึ่งอักษร)
- Variant (16 ไบต์ + 1 ไบต์ต่อหนึ่งอักษร)

User-defined (จำนวนไบต์ขึ้นอยู่กับตัวแปรประกอบ ตัวแปรประเภทนี้จะมีตัวแปรย่อยอื่นภายในตัวมัน ทำนองเดียวกันกับหนึ่งเรคอร์ดมีฟิลด์หลายฟิลด์)

นอกจากนี้คุณยังสามารถใช้ตัวแปรเป็นอะเรย์ได้อีกด้วย การดีแคลร์ตัวแปรอะเรย์ทำได้โดยการระบุชื่อตัวแปรแล้วตามด้วยไดเมนชันของอะเรย์ในวงเล็บ อย่างเช่น

```
Dim Int_numbers (10, 10) As Integer
```

สเตทเมนต์ที่เห็นหมายถึง การดีแคลร์อะเรย์สองมิติเป็น 10x10 โดยมีชนิดเป็นเลขจำนวนเต็ม การใช้อะเรย์เป็นไปได้ถึง 60 มิติ ลองนึกดูเถอะครับว่าถ้ามีใครใช้ได้มันจะเป็นอย่างไร

เราเห็นกันบ่อยครับที่จะใช้ตัวแปรมารับค่าพรอพเพอร์ตี้ส์หรือค่ากำหนดของออปเจ็กต์ อย่างเช่น คุณต้องการแสดงเนื้อความที่อยู่ใน Range ("A1") ในเมสเสจบ็อกซ์ ก็อาจทำได้ดังนี้

```
Sub Display_A1()
        Dim Int_one As Integer
        Int_one = Range ("A1").Value
        MsgBox Int_one
End Sub
```

มีตัวแปรอยู่สามชนิดที่ทรงความสามารถสูง นั่นคือ ตัวแปรชนิด object, variant, และ user-defined ตัวแปรออปเจ็กต์มีประโยชน์มากเกินในแง่ที่มันถูกใช้แทนออปเจ็กต์ 128 ตัวในเอ็กเซลได้ทั้งสิ้น และการที่จะกำหนดให้ตัวแปรใดมีค่าเป็นออปเจ็กต์ใดก็ง่ายมาก เพียงใช้คีย์เวิร์ด Set เท่านั้น ข้างล่างนี้จะเป็นตัวอย่างที่แสดงให้เห็นว่าตัวแปรออปเจ็กต์ถูกใช้ในการป้อนค่า ("Hello") เข้าไปในเรนจ์ A1 บนเวอร์กชีตชื่อ Sheet1 ในเวอร์กบุ๊คชื่อ Book1.xls ได้อย่างไร

```
Sub Assign_A1 ()
        Dim cella1 As Object
        Set cella1 = Workbooks
("Book1.xls").Worksheets("Sheet1").
Range("A1")
        cella1.Value = "Hello"
End Sub
```

สังเกตนะครับว่าถ้าเราต้องป้อนค่าตรง ๆ มันยาวมาก แต่ถ้ากำหนดให้ cella1 มาแทนไอ้ที่มันยาว ๆ นั้นเสียเวลาป้อนค่าก็จะกะทัดรัด ดูง่าย เขียนง่ายขึ้นกว่าเดิมมาก มันดีอย่างนี้แหละ อย่าลืมว่าการเขียนโค้ดจริงคุณอาจต้องการเปลี่ยนแปลงแก้ไข cella1.Value หลายจุด หากไม่มีวิธีลัดอย่างนี้คุณก็จะเปลืองแรงเขียนสิ่งที่ยืดยาวและกลับมาอ่านยากในภายหลังอีกด้วย

นอกจากนี้ยังมีคุณสมบัติที่แสนวิเศษของตัวแปรชนิด variant เข้ามาเพิ่มความง่ายในการโค้ดเข้าไปอีก ตัวแปรชนิดนี้เมื่อดีแคลร์แล้วคุณสามารถป้อนค่าอะไรให้มันก็ได้ ไม่ต้องพะวงเรื่องชนิดข้อมูลให้เสียเวลา คุณไม่จำเป็นต้องรู้ล่วงหน้าเลยว่ากำลังจะใช้มันทำอะไร เมื่อไหร่ที่ไหน อย่างเช่น สมมติว่ามีค่าอะไรอย่างหนึ่งอยู่ในเซล A1 แต่ไม่แน่ไม่นอนว่าชนิดข้อมูลมันเป็นอะไร เราเขียนโค้ดได้ว่า

```
Sub Get_A1 ()
        Dim cella1 (การดีแคลร์
ให้เป็น variant ทำได้โดยคุณไม่ระบุ
ชนิดอย่างนี้)
        cella1 = Range("A1").
Value
        MsgBox cella1
End Sub
```

สังเกตนะครับว่าคุณ Dim แล้วตามด้วยชื่อตัวแปรที่ต้องการลอย ๆ ก็กลายเป็น variant ไปแล้วโดยอัตโนมัติ

ตัวแปรอีกชนิดหนึ่งที่น่ากล่าวขวัญถึงประโยชน์ของมันคือ ชนิด user-defined ตัวแปรชนิดนี้ต้องทำการกำหนดนอกโพรซีเดอร์ครับ เมื่อดีแคลร์ต้องใช้คีย์เวิร์ด Type โดยเราจะดีแคลร์ที่ส่วนบนสุดของโมดูลเขียนโค้ดก่อนโพรซีเดอร์อื่นใดทั้งสิ้น ตัวอย่างข้างล่างนี้แสดงการดีแคลร์ตัวแปรชนิด user-defined ซึ่งจะนำมาแสดงในเมสเสจบ็อกซ์

```
Type MyData
        MyName As String
        MyAge As Integer
End Type

Sub Person_Data ()
        Dim Person_x As MyData
        Person_x.MyName = John
        Person_x.MyAge = 23
        MsgBox Person_x.MyName
& ", " & Person_x.MyAge
End Sub
```

ตัวแปรต่าง ๆ จะมีขอบเขตการใช้ในระดับต่าง ๆ ขึ้นอยู่กับว่ามันถูกดีแคลร์ไว้ที่ไหน คือตัวแปรที่ดีแคลร์ในโพรซีเดอร์จะมีขอบเขตในการใช้เฉพาะในโพรซีเดอร์นั้น ๆ เราเรียกขอบเขตนี้ว่า procedure-level ส่วนตัวแปรที่ดีแคลร์ไว้ ณ ส่วนบนสุดของโมดูลก่อนโพรซีเดอร์อื่นใด จะสามารถใช้ได้ทุกโพรซีเดอร์ในโค้ด เรียกขอบเขตนี้ว่า module-level

สำหรับผู้ที่เคยใช้วิชวลเบสิกมาก่อนคงจินตนาการออก ว่าส่วนบนสุดของโค้ดนี้ก็เหมือนส่วนประกาศเฉพาะ (declaration section) นั่นเอง แต่สำหรับเอ็กเซลมีเรื่องแตกต่างหน่อยหนึ่งคือ ถ้าคุณต้องการให้มันใช้ได้ทั่วทั้งโปรเจ็กต์ เราต้องใช้คำว่า Public แทนคำว่า Global

โครงสร้างการควบคุมในโปรแกรม

คราวนี้มาว่าถึงคำสั่งควบคุมการปฏิบัติการต่าง ๆ ซึ่งก็มีทั้งการใช้เงื่อนไขและการกำหนดให้วนลูปคือ If-Then-Else, Select Case, For-Next, และ Do-While-Loop โครงสร้างเหล่านี้ดำเนินการในวิชวลเบสิกสำหรับแอพพลิเคชันในวิถีทางเดียวกันกับที่ดำเนินการกันในโปรแกรมอื่นทั้งหลายนั่นเอง ลองมาดูสักสองสามตัวอย่าง

If-Then-Else

แมคโครที่ยกเป็นตัวอย่างข้างล่างใช้โครงสร้างโค้ดแบบที่ว่าถ้าสิ่งหนึ่งเกิดขึ้นจะให้ทำอย่างไรต่อไป และถ้ามันไม่เกิดขึ้นจะให้ทำอะไรแทน โดยผมจะสมมติว่าเราต้องการดูเนื้อความใน Range ("A1") มันเป็นคำว่า "True" หรือไม่ ถ้าใช่ก็ให้แสดงเมสเสจบ็อกซ์บอกว่าถูก ถ้าไม่ใช่ก็ให้บอกว่าผิด

```
Sub Test_Answer ()
        Dim var_one as String

        var_one = Range("A1").Value

        If var_one = "True" Then
            MsgBox "ใช่เลย"
        Else
            MsgBox "มิใช่เช่นนั้น"
        End If
    End Sub
```

Select Case

อันนี้เป็นกลไกการควบคุมการดำเนินการที่คล้ายคลึงกับ If-Then-Else ต่างกันที่วิธีปฏิบัติต่อเงื่อนไข เรามาดูตัวอย่างการใช้ฟังก์ชัน InputBox ประกอบเพื่อขอรหัสผ่านจากผู้ใช้กัน งานนี้เรามีเงื่อนไขอยู่สามอย่างคือ ผู้ใช้ป้อนรหัสผ่านที่ 1 ถูก จะได้สิทธิ์ทั้งอ่านและเขียนข้อมูล ถ้าผู้ใช้ป้อนรหัสผ่านที่ 2 ถูก จะได้สิทธิ์เฉพาะอ่านข้อมูลอย่างเดียว แต่ถ้าไม่สามารถป้อนทั้งรหัสแรกและรหัสที่สองก็หมดสิทธิ์เข้าถึงแอพพลิเคชันอย่างสิ้นเชิง

```
Sub Enter_Password ()
        PassWord = InputBox
("Enter Password:", "Password")
        Select Case PassWord
        Case "levelone"
            MsgBox "คุณได้
สิทธิ์ทั้งในการอ่านและเขียน"
        Case "leveltwo"
            MsgBox "คุณได้
สิทธิ์ในการอ่านอย่างเดียว"
        Case Else
            MsgBox "คุณไม่มี
สิทธิ์ใช้แอพพลิเคชันนี้หรอก จะบอกให้"
        End Select
    End Sub
```

For-Next

เราใช้กลไกการควบคุมชนิดนี้เพื่อให้เกิดการทำซ้ำครับ อย่างเช่น ต้องการให้แมคโครมันอ่านค่าจากเซล A1 จากนั้นก็ใช้ลูป For-Next ในการคำนวณแฟคทอเรียลของเลข และแสดงค่าที่ได้ออกมาทางเมสเสจบ็อกซ์

```
Sub Find_Factorial ()
    Dim var_one As Integer
    Dim var_two As Double
    var_one = Range("A1").Value
    var_two = 1
    For x = var_one To 1 Step -1
        var_two = var_two * x
    Next x
    MsgBox "แฟคทอเรียลของ"
& var_one & " คือ " & var_two
End Sub
```

Do-While

คล้ายกับ For-Next นั่นแหละครับ มีข้อยกเว้นคือ แทนที่จะระบุจำนวนลูปที่แน่นอน เราใช้เงื่อนไขอะไรสักอย่างมาเป็นตัวกำหนดแทนว่า หากเงื่อนไขนั้นเป็นจริงจึงจะให้หยุดปฏิบัติงานในลูป ตัวอย่างข้างล่างนี้เป็นการแสดงอินพุตบ็อกซ์ซึ่งพร้อมรับการป้อนข้อมูลจากผู้ใช้ โดยที่มันจะหยุดแสดงต่อเมื่อผู้ใช้ใส่คำว่า "กรุงเทพฯ" ลงไปเป็นอินพุต

```
Sub Geo_Quiz ()
    Dim province_name As String
    province_name = ""
    Do While province_name <>
"กรุงเทพฯ"
        province_name = InputBox
("ใส่ชื่อจังหวัด:", "Geopgraphy Quiz")
    Loop
End Sub
```

มีโครงสร้างโค้ดแมคโครอีกอย่างเดียวที่มีความเป็นสามัญน่ารู้ไว้คือ With ซึ่งจะอนุญาตให้คุณใช้ตัวย่อแทนออปเจ็กต์หนึ่งในการกำหนดพรอพเพอร์ตี้สำจำนวนมากให้กับมัน อย่างเช่น คุณต้องการเซตฟอนต์ให้กับเรนจ์ A1 โดยเจาะจงให้มีขนาด 12 ชื่อฟอนต์เป็น Times และสร้างฟอนต์แบบตัวหนากับเอียงพร้อมกัน ลองมาดูเปรียบเทียบกัน เอาแบบไม่ใช้ With ก่อน

```
Sub Set_Font ()
        Worksheets("sheet1").
Range("a1").Font.Size = 12
        Worksheets("sheet1").
Range("a1").Font.Name = "Times"
        Worksheets("sheet1").
Range("a1").Font.Bold = True
        Worksheets("sheet1").
Range("a1").Font.Italic = True
End Sub
```

แล้วคราวนี้ลองแบบใช้ With ดูบ้าง

```
Sub Set_Font ()
        With Worksheets
("sheet1").Range("a1").Font
            .Size = 12
            .Name = "Times"
            .Bold = True
            .Italic = True
        End With
End Sub
```

จะเห็นว่าสบายกว่ากันเยอะเลย แถมอ่านก็ง่ายกว่าด้วย สำหรับการย่อในลักษณะนี้คุณที่เป็นวิชวลเบสิกมาก่อนไม่ต้องไปค้นคว้าให้เสียเวลาครับ เพราะมีเฉพาะในเอ็กเซลเท่านั้น

นอกจากจุดสำคัญที่กล่าวถึงแล้วข้างต้น ยังมีเรื่องรองที่น่าสนใจอยู่อีกสองเรื่อง ถือว่าเป็นของแถมก็แล้วกันคือ กระบวนการในการเรียกแมคโครหนึ่งจากอีกแมคโครหนึ่ง และกระบวนการในการผ่านอาร์กิวเมนต์ระหว่างแมคโคร

เพื่อที่จะเรียกแมคโครหนึ่งจากอีกแมคโครหนึ่ง คุณเพียงรวมชื่อของแมคโครที่ต้องการเรียกเข้าไว้ในรูปคำสั่ง ดังตัวอย่างข้างล่างนี้

```
Sub Macro_One ()
    Do While (InputBox("ใส่ชื่อจังหวัด:", "Geography Quiz") <> "กรุงเทพฯ")
    Loop
    Display_Message
End Sub

Sub Display_Message ()
    MsgBox "คำตอบของคุณถูกต้องแล้ว"
End Sub
```

คำสั่งแมคโครข้างต้นใช้ Do-While มาเป็นตัวกำหนดแสดงกล่องคำถามเพื่อให้ผู้ใช้ป้อนคำตอบ ถ้าเมื่อไหร่ที่ใส่ "กรุงเทพฯ" ก็เป็นอันว่าแมคโครแรกจะหลุดจากลูป DoWhile ทันที จากนั้นก็จะปฏิบัติตามคำสั่งบรรทัดถัดจาก Loop ซึ่งก็ไปพบคำว่า Display_Message มันก็จะรู้เลยว่าคุณต้องการให้มันเรียกแมคโคร Display_Message ซึ่งโค้ดในส่วนแมคโครนั้นก็จะสั่งให้เกิดการแสดงเมสเสจบ็อกซ์บอกกล่าวว่า "คำตอบของคุณถูกต้องแล้ว" ในที่นี้ลองสังเกตวิธีใช้เครื่องหมาย <> ในการเปรียบเทียบคำตอบ "กรุงเทพฯ" ด้วยนะครับ

ตัวอย่างต่อไปเป็นการแสดงอินพุตบ็อกซ์เพื่อเอาค่าอาร์กิวเมนต์จากผู้ใช้

```
Sub Macro_Two ()
    Dim Province_Name As String
    Province_Name = InputBox("ใส่ชื่อจังหวัด:", "GeoQuiz")
    Display_Answer (Province_Name)
End Sub

Sub Display_Answer (Province_Name)
    MsgBox "จังหวัดที่คุณใส่มาคือ" & Province_Name
End Sub
```

ในที่นี้ Macro_Two ใช้ตัวแปร Province_Name เพื่อเก็บค่าสตริงที่ผู้ใช้ป้อนผ่านอินพุตบ็อกซ์มา แล้วบรรทัดสุดท้ายผ่านตัวแปรนี้ไปให้แมคโคร Display_Answer ซึ่งจะไม่มีการเช็คอะไรทั้งสิ้น ส่งมาอย่างไรก็จะแสดงให้อย่างนั้นเลย

ณ ที่นี้ขอให้สังเกตเครื่องหมายที่ใช้กันมาบ่อยแล้วคือ & ขอบอกให้ทราบสำหรับบางคนที่อาจยังไม่รู้ว่าเป็นเครื่องหมายที่จะเชื่อมสตริงมากกว่าหนึ่งตัวเข้าด้วยกัน คุณอาจใช้เครื่องหมาย + แทนก็ได้นะครับ แต่ในทางปฏิบัติมันจะดูดี ดูเข้าใจง่ายกว่า ว่าเราจำเพาะให้เป็นการรวมตัวอักษรหรืออักขระพิเศษที่ไม่ใช่การบวกลบทางคณิตศาสตร์

โดยสรุปแล้ววิชวลเบสิกสำหรับแอพพลิเคชันให้เครื่องไม้เครื่องมือในการทำงานกับออปเจ็กต์ของเอ็กเซลมามาก มันเป็นหนทางที่คุณจะเข้าถึงและกำหนดค่าพรอพเพอร์ตี้ส์ให้กับออปเจ็กต์ รวมทั้งการเรียกใช้พวกมันโดยประการต่าง ๆ พิสดารไปเรื่อย ๆ สุดแต่คุณจะเสกสรรค์

2. ออปเจ็กต์ต่าง ๆ ของเอ็กเซล

ทีนี้ก็มาถึงเรื่องที่ทุกคนไม่ว่าหน้าเก่าหรือใหม่ในงานโค้ดเชิงวัตถุอยากทราบ นั่นคือ รายละเอียดเกี่ยวกับออปเจ็กต์ของเอ็กเซล

อย่างที่เคยกล่าวมาแล้วว่าออปเจ็กต์ต่าง ๆ ถูกจัดเรียงแบบลำดับขั้น (Hierarchy Order) ออปเจ็กต์ซึ่งเป็นโกลบอลมากที่สุดคือ Application ไม่มีอะไรใหญ่ ไม่มีลำดับไหนอยู่สูงกว่านี้อีกแล้ว เนื่องจากเป็นออปเจ็กต์ที่ใช้แทนเอ็กเซลทั้งตัวนั่นเอง ไดอะแกรมของออปเจ็กต์โมเดลตามรูปที่ 1 นี้ใช้แทนกลุ่ม

ออปเจ็กต์สำคัญที่กว้างที่สุด มันเป็นธรรมชาติของลำดับออปเจ็กต์ในเอ็กเซล

มันเป็นเรื่องสำคัญที่เราต้องทราบว่ามีออปเจ็กต์อยู่หลายตัวในเอ็กเซลที่ถูกเรียกขานว่า คอลเลคชัน (Collections) โดยที่คอลเลคชันหนึ่งก็ไม่ใช่อะไรอื่นนอกจากกลุ่มออปเจ็กต์ที่เหมือน ๆ กัน อย่างเช่น ออปเจ็กต์เวอร์กบุ๊คนี่ก็เป็นคอลเลคชัน มันใช้แทนไฟล์เวอร์กบุ๊คที่ถูกเปิดใช้ทั้งหมด (ทำนองเดียวกันกับที่ Forms ในวิชวลเบสิกเป็นคอลเลคชันของฟอร์มทั้งหมดที่ถูกเปิดใช้อยู่นั่นแหละครับ)

คอลเลคชันมีประโยชน์มาก เพราะมันให้ความสามารถโปรแกรมเมอร์ในการทำงานอย่างเดียวกันกับออปเจ็กต์ทั้งหมดอย่างทั่วถึง อย่างเช่น คุณกำลังต้องการเขียนแมคโครที่จะปิดเวอร์กบุ๊คทั้งหมดซึ่งกำลังแอคตีฟอยู่ในเอ็กเซล ถ้าหากเป็นสมัยก่อนคุณต้องเขียนคำสั่งแมคโครเพื่อดูให้แน่ว่าตอนนั้นมีไฟล์ที่ถูกเปิดขึ้นอยู่ทั้งหมดเท่าไหร่ แต่ละไฟล์ชื่ออะไรบ้าง จากนั้นก็ต้องตระเวนปิดทีละไฟล์เปลืองโค้ดไปโดยใช่เหตุหลายต่อหลายบรรทัด

แต่เมื่อมาถึงสมัยของการใช้คอลเลคชัน คุณแค่ทำง่าย ๆ เพียงเขียนโค้ดในรูป For-Each อันจะทำให้เกิดการทำงานตลอดสายคอลเลคชันดังตัวอย่างข้างล่างนี้

```
Sub Close_Workbooks ()
    For Each Wrk_book In Appli
cation.Workbooks
        Wrk_book.close
    Next
End Sub
```

เห็นไหมครับว่าคุณไม่จำเป็นต้องรู้ชื่อแซ่ไฟล์เวอร์กบุ๊คอะไรเลย การใช้ For-Each ทำให้มันรู้อยู่แล้วว่าหมายถึงสิ่งที่กำลังปรากฏอยู่ในหน่วยความจำ ที่เห็นในไดอะแกรมนั้นทุกตัวที่เป็นแถบมืดล้วนแล้วแต่เป็นคอลเลคชันทั้งสิ้น นั่นหมายความว่าต่อไปนี้คุณมีเครื่องทุ่นแรงที่ช่วยให้คุณเบาภาระ

ในไดอะแกรมที่เห็น ส่วนแรกจะเป็นแก่นของโมเดล ส่วนที่สองเป็นออปเจ็กต์ที่อยู่ใต้อาณัติของออปเจ็กต์ชื่อ DrawingObjects ซึ่งเป็นสัปออปเจ็กต์ของออปเจ็กต์เวอร์กชีต ชาร์ต และไดอะล็อกชีต

ดังที่กล่าวไว้แล้ว เมื่อจะเขียนแมคโครในเอ็กเซล คุณต้องผ่าน "ลง" ไปตามลำดับขั้นด้วยวิธีใช้จุด (.) เป็นตัวแยก อย่างเช่น ต้องการแทรกสตริง "สวัสดีจ้า!" เข้าไปในช่องเซลหนึ่งบนเวอร์กชีตก็จะเห็นโค้ดเหมือนอย่างนี้

```
Sub Say_Hello()
    Application. Workbook.Worksheet.
Range.Value = "สวัสดีจ้า!"
End Sub
```

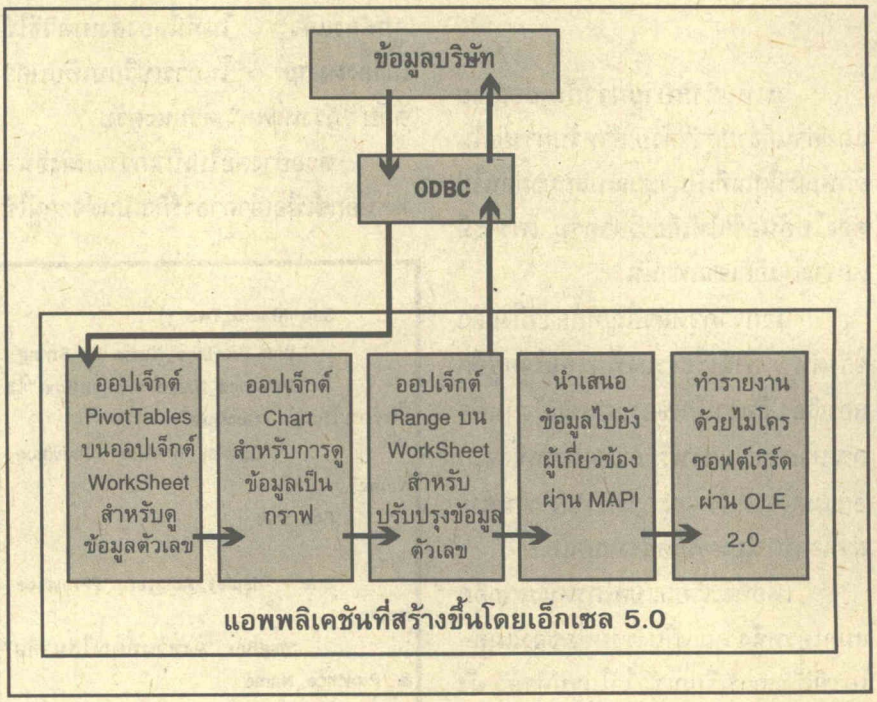
ในทำนองเดียวกันนี้หาก Sheet1 กำลังทำงาน การอ้างอิงถึงสามารถเปลี่ยนเป็น ActiveSheet ได้

```
Sub Say_Hello ()
    ActiveWorkbook.ActiveSheet.
Range ("A1").Value = "สวัสดีจ้า!"
End Sub
```

และหาก Sheet1 ใน Book1.xls กำลังทำงาน โค้ดนี้สามารถย่อให้สั้นเข้ามาได้อีก

```
Sub Say_Hello ()
    Range ("A1").Value = "สวัสดีจ้า!"
End Sub
```

สมมติว่าเรนจ์ A1 ถูกคลิ๊กเลือกแล้วเรียบร้อย คุณก็ใช้คำแทนสั้น ๆ อีกอย่างได้โดยไม่ต้องระบุชื่อมันเลยว่า



รูปที่ 1 แสดงไดอะแกรมการนำ EXCEL 5.0 มาใช้สร้างแอพพลิเคชัน

```
Sub Say_Hello ()
        Selection.Value  =  "สวัสดีจ้า!"
End Sub
```

เพราะอย่างนั้น โค้ดทั้งหมดข้างล่างนี้จึงทำงานอย่างเดียวกันครับ

```
        Application.  Workbook("Book1.
xls").Worksheet(Sheet1").Range("A1").
Value  =  "สวัสดีจ้า!"
        Application.  Workbook.Worksheet.
Range.Value  =  "สวัสดีจ้า!"
        ActiveWorkbook.ActiveSheet.
Range  ("A1").Value  =  "สวัสดีจ้า!"
        ActiveSheet.Range  ("A1").Value
 =  "สวัสดีจ้า!"
        Range  ("A1").Value  =  "สวัสดีจ้า!"
        Selection.Value  =  "สวัสดีจ้า!"
```

ความแตกต่างกับพรอพเพอร์ตี้ส์ประการหนึ่งก็คือ เมธอดไม่มีการกำหนดค่า ข้อนี้คุณจะรู้ชัดไปเองเมื่อใช้วิชวลเบสิกจนคล่อง พอพูดถึงการใช้เมธอดคุณจะนึกถึงการสั่งให้ออปเจ็กต์สักตัวปฏิบัติอะไรสักอย่างทันที สิ่งที่มันต้องการคือค่าอาร์กิวเมนต์เฉพาะตัว ค่าอาร์กิวเมนต์นี้ต้องให้หรือไม่ให้ก็ต้องดูว่าคุณจะใช้อะไรและอย่างไร มันจะเป็นตัวกำหนดการปฏิบัติตามคำสั่งนั้น ๆ ว่าจะเป็นไปในทิศทางใด ตัวอย่างที่ดีน่าจะได้แก่การ "ปิด" ออปเจ็กต์เวอร์กบุ๊คครับ เมธอด Close จะถูกใช้เพื่องานนี้ และสามารถกำหนดวิธีการปิดได้พร้อมสามทางเลือกคือ

- SaveChanges: ถ้าเป็น True ไฟล์เวอร์กบุ๊คนั้นจะถูกเซฟก่อนปิด
- FileName: ให้คุณกำหนดว่าจะเซฟเป็นไฟล์ชื่ออะไรก่อนปิด
- teWrkbook: ถ้าเป็น True เวอร์กบุ๊คที่จะถูกปิดนี้จะเข้าไปอยู่ในลิสต์รายชื่อให้ผู้ใช้เลือก

อาร์กิวเมนต์ทั้งสามข้างบนนี้คุณสั่งได้ว่าจะเลือกหรือไม่เลือก ดังนั้นมันจึงเป็นไปได้ที่จะปิดเวอร์กบุ๊คด้วยเมธอด Close ดังตัวอย่างข้างล่าง

```
Workbooks  ("Book1.xls").Close
```

คำสั่งข้างบนนี้จะทำให้เอ็กเซลขอให้ผู้ใช้ทำการเซฟ (ไม่ใช่เซฟให้เองนะครับ) และเมธอด Close ก็มีรูปแบบที่สามารถเรียกพารามิเตอร์ตามการสั่งเลือกจากโค้ดได้อีกด้วย อย่างเช่น ผมอยากจะให้เอ็กเซลมันเซฟเป็นไฟล์ชื่อ Book2.xls ให้โดยอัตโนมัติก่อนที่จะทำการปิด และไม่ต้องการให้เวอร์กบุ๊คอยู่ในลิสต์ของผู้ใช้คนต่อไป ก็จะเขียนอย่างนี้ครับ

```
        Workbooks("book1.xls").Close
savechanges:=True,  filename:="Book2.
xls",  routeworkbook:=False
```

จากตัวอย่างที่เห็นนี้สังเกตได้ว่าการกำหนดค่าจะต้องทำด้วยเครื่องหมาย := มันเป็นไปได้ที่จะผ่านค่าไปให้อาร์กิวเมนต์โดยไม่ต้องอ้างชื่ออาร์กิวเมนต์ ถ้าหากค่าของมันถูกวางไว้ตรงตามลำดับเช่นที่เห็น การใช้ชื่ออาร์กิวเมนต์พร้อมเครื่องหมาย := จะทำให้คุณไม่ต้องกังวลเกี่ยวกับเรื่องของลำดับ แล้วก็ไม่จำเป็นต้องผ่านค่าไปทั้งหมดทุกตัว อย่างเช่น ผมพอใจจะเขียนเสียใหม่ตามแต่จะนึกได้ว่า

```
        Workbooks("book1.xls").Close
filename:="Book2.xls",  routeworkbook:=
False,  savechanges:=True
```

หากว่าคุณผ่านค่าไปยังอาร์กิวเมนต์ในเมธอดหนึ่งแต่ไม่มีชื่อของอาร์กิวเมนต์กำกับ คุณก็ทำได้แต่ต้องแน่ใจนะว่าลำดับต้องตรงตามกฎ ตัวอย่างข้างล่างเป็นการปิดเวอร์กบุ๊คเหมือนตัวอย่างข้างบนโดยไม่ต้องมีชื่ออาร์กิวเมนต์เข้ามาเกะกะ ไมโครซอฟต์เขารู้ครับว่ามืออาชีพที่คุ้นกับวิถีทางซ้ำ ๆ เป็นเดือนเป็นปีนั้นต้องการอะไรที่ลัด ๆ กันทุกคน

```
        Workbooks("book1.xls").Close
True,  "book2.xls",  False
```

นี่เป็นการแสดงให้เห็นเป็นอย่างดีใช่ไหมครับว่าวิชวลเบสิกสำหรับแอพพลิเคชันของไมโครซอฟต์เขาเด็ดขนาดไหน ไม่ว่าจะเป็นการกำหนดค่าให้พรอพเพอร์ตี้ส์หรือเซตอาร์กิวเมนต์ให้เมธอด มันง่ายและทรงสมรรถนะไปหมด เอาล่ะครับเราคงพอสัมผัสและสำเหนียกกันแล้วว่าออปเจ็กต์แบบลำดับชั้นนั้นมันหยิบจับใช้สอยกันได้อย่างไร คราวต่อไปจะนำไปสัมผัสกับออปเจ็กต์ที่น่าสนใจบางตัวในเอ็กเซลกับพรอพเพอร์ตี้ส์และเมธอดที่เกี่ยวข้อง สวัสดีครับ